

## **UNIT II PROGRAMMING OF 8085 PROCESSOR**

Instruction format and addressing modes – Assembly language format – Data transfer, data manipulation & control instructions – Programming: Loop structure with counting & Indexing - Look up table - Subroutine instructions - stack.

**Instruction Format in 8085****Instruction**

- Command to the microprocessor to perform a given task on a Specified data.

Two parts

**operation code** (opcode) - task to be performed

**operand** - data to be operated on.

**Instruction word size**

Three groups

1. One-word or 1-byte instructions
2. Two-word or 2-byte instructions
3. Three-word or 3-byte instructions

**One-Byte Instructions**

- Includes the opcode and operand in the same byte.
- requires one memory location

Ex.

**MOV C, A** - Copy the contents of the accumulator in the register C.

**ADD B** - Add the contents of register B to the contents of the acc

**CMA** - Invert (complement) each bit in the accumulator

**Two-Byte Instructions**

- first byte - the operation code
- Second byte - operand
- Source operand is a data byte immediately following the opcode.
- Require two memory locations to store in memory

Ex.

**MVI A, Data** - Load an 8-bit data byte in the accumulator

**ADI ,data** -  $A \leftarrow A + \text{data}$

**Three-Byte Instructions**

first byte - opcode

- Following two bytes specify the 16-bit address.
- second byte is the low-order address
- Third byte is the high-order address.
- require three memory locations to store in memory.

Ex.

**JMP 2085H** - Transfer the program sequence to the memory location 2085H.

**LXI rp, data16** - rp is one of the pairs of registers BC, DE, HL used as 16-bit register.

### Addressing modes of 8085 Microprocessor

The method by which the address of source of data or the address of destination of result is given in the instruction is called addressing modes.

The term addressing mode refers to the way in which the operand of the instruction is specified.

Types of Addressing Modes

1. Direct Addressing Mode
2. Register Addressing Mode
3. Register Indirect Addressing Mode
4. Immediate Addressing Mode
5. Implicit Addressing Mode

#### **Direct Addressing Mode**

In this mode, the address of the operand is given in the instruction itself.

**LDA 2500 H** Load the contents of memory location 2500 H in accumulator

LDA is the operation.

2500 H is the address of source.

Accumulator is the destination.

#### **Register Addressing Mode**

In this mode, the operand is in general purpose register.

**MOV A, B** Move the contents of register B to A

MOV is the operation.

B is the source of data.

A is the destination.

#### **Register Indirect Addressing Mode**

In this mode, the address of operand is specified by a register pair.

**MOV A, M** Move data from memory location specified by H-L pair to accumulator.

MOV is the operation.

M is the memory location specified by H-L register pair.

A is the destination.

#### **Immediate Addressing Mode**

In this mode, the operand is specified within the instruction itself.

**MVI A, 05 H** Move 05 H in accumulator.

MVI is the operation.

05 H is the immediate data (source).

A is the destination.

### Implicit Addressing Mode

If address of source of data as well as address of destination of result is fixed, then there is no need to give any operand along with the instruction.

**CMA**            Complement accumulator.

CMA is the operation.

A is the source.

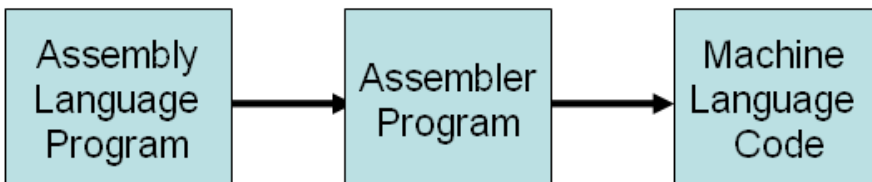
A is the destination.

### Assembly Language format of 8085

- A microprocessor executes instructions given by the user
- Instructions should be in a language known to the microprocessor
- Microprocessor understands the language of 0's and 1's only
- This language is called Machine Language .

### Assembly Language of 8085

- It uses English like words to convey the action/meaning called as MNEMONICS
- For e.g.
  - MOV            to indicate data transfer
  - ADD            to add two values
  - SUB            to subtract two values
- Microprocessor cannot understand a program written in Assembly language
- A program known as **Assembler** is used to convert a Assembly language program to machine language



### Low-level/High-level languages

- Machine language and Assembly language are both
  - Microprocessor specific (Machine dependent) are called low-level languages
- Machine independent languages are called High-level languages
  - For e.g. BASIC, PASCAL,C++,C,JAVA, etc.

- A software called Compiler is required to convert a high-level language program to machine code

### **Instruction set of 8085.**

- An instruction is a binary pattern designed inside a microprocessor to perform a specific function.
- The entire group of instructions that a microprocessor supports is called *Instruction Set*.
- 8085 has **246** instructions.
- Each instruction is represented by an 8-bit binary value.
- These 8-bits of binary value is called *Op-Code* or *Instruction Byte*.

The 8085 instructions can be classified into the following five functional categories:

1. Data transfer (copy) group
2. Arithmetic group
3. Logical group
4. Branching group and
5. I / O and Machine control group

### **Data Transfer Group:**

- These instructions move data between registers, or between memory and registers.
- These instructions copy data from source to destination.
- While copying, the contents of source are not modified.

### **Instructions**

**r – General Purpose register (B, C, D, E, H, L)**

**rp- register pair (BC, DE, HL)**

- MOV r1, r2** (Move Data; Move the content of the one register to another).  $[r1] \leftarrow [r2]$ .
- MOV r, M** (Move the content of memory register).  $r \leftarrow [M]$
- MOV M, r.** (Move the content of register to memory).  $M \leftarrow [r]$
- MVI r, data.** (Move immediate data to register).  $[r] \leftarrow \text{data}$ .
- MVI M, data.** (Move immediate data to memory).  $M \leftarrow \text{data}$ .
- LXI rp, data 16.** (Load register pair immediate).  $[rp] \leftarrow \text{data 16 bits}$ ,  $[rh] \leftarrow 8 \text{ LSBs of data}$ .
- LDA addr.** (Load Accumulator direct).  $[A] \leftarrow [\text{addr}]$ .
- STA addr.** (Store accumulator direct).  $[\text{addr}] \leftarrow [A]$ .
- LHLD addr.** (Load H-L pair direct).  $[L] \leftarrow [\text{addr}]$ ,  $[H] \leftarrow [\text{addr}+1]$ .
- SHLD addr.** (Store H-L pair direct)  $[\text{addr}] \leftarrow [L]$ ,  $[\text{addr}+1] \leftarrow [H]$ .
- LDAX rp.** (LOAD accumulator indirect)  $[A] \leftarrow [[rp]]$

**l. STAX rp.** (Store accumulator indirect)  $[[rp]] \leftarrow [A]$ .

**m. XCHG.** (Exchange the contents of H-L with D-E pair)  $[H-L] \leftrightarrow [D-E]$ .

### Arithmetic Group:

Perform arithmetic operations

- Addition,
- Subtraction;
- Increment or decrement of the content of a register or memory.

- i. **ADD r.** (Add register to accumulator)  $[A] \leftarrow [A] + [r]$ .
- ii. **ADD M.** (Add memory to accumulator)  $[A] \leftarrow [A] + [[H-L]]$ .
- iii. **ADC r.** (Add register with carry to accumulator).  $[A] \leftarrow [A] + [r] + [CS]$ .
- iv. **ADC M.** (Add memory with carry to accumulator)  $[A] \leftarrow [A] + [[H-L]] [CS]$ .
- v. **ADI data** (Add immediate data to accumulator)  $[A] \leftarrow [A] + \text{data}$ .
- vi. **ACI data** (Add with carry immediate data to accumulator).  $[A] \leftarrow [A] + \text{data} + [CS]$ .
- vii. **DAD rp.** (Add register pair to H-L pair).  $[H-L] \leftarrow [H-L] + [rp]$ .
- viii. **SUB r.** (Subtract register from accumulator).  $[A] \leftarrow [A] - [r]$ .
- ix. **SUB M.** (Subtract memory from accumulator).  $[A] \leftarrow [A] - [[H-L]]$ .
- x. **SBB r.** (Subtract register from accumulator with borrow).  $[A] \leftarrow [A] - [r] - [CS]$ .
- xi. **SBB M.** (Subtract memory from accumulator with borrow).  $[A] \leftarrow [A] - [[H-L]] - [CS]$ .
- xii. **SUI data.** (Subtract immediate data from accumulator)  $[A] \leftarrow [A] - \text{data}$ .
- xiii. **SBI data.** (Subtract immediate data from accumulator with borrow).
  - i.  $[A] \leftarrow [A] - \text{data} - [CS]$ .
- xiv. **INR r** (Increment register content)  $[r] \leftarrow [r] + 1$ .
- xv. **INR M.** (Increment memory content)  $[[H-L]] \leftarrow [[H-L]] + 1$ .
- xvi. **DCR r.** (Decrement register content).  $[r] \leftarrow [r] - 1$ .
- xvii. **DCR M.** (Decrement memory content)  $[[H-L]] \leftarrow [[H-L]] - 1$ .
- xviii. **INX rp.** (Increment register pair)  $[rp] \leftarrow [rp] + 1$ .
- xix. **DCX rp** (Decrement register pair)  $[rp] \leftarrow [rp] - 1$ .
- xx. **DAA** (Decimal adjust accumulator) .

### Logical Group:

Perform logical operation such as

- AND, OR, XOR
  - Compare,
  - Rotate etc.
- i. **ANA r.** (AND register with accumulator)  $[A] \leftarrow [A] \wedge [r]$ .

- ii. **ANA M.** (AND memory with accumulator).  $[A] \leftarrow [A] \wedge [[H-L]]$ .
- iii. **ANI data.** (AND immediate data with accumulator)  $[A] \leftarrow [A] \wedge \text{data}$ .
- iv. **ORA r.** (OR register with accumulator)  $[A] \leftarrow [A] \vee [r]$ .
- v. **ORA M.** (OR memory with accumulator)  $[A] \leftarrow [A] \vee [[H-L]]$
- vi. **ORI data.** (OR immediate data with accumulator)  $[A] \leftarrow [A] \vee \text{data}$ .
- vii. **XRA r.** (EXCLUSIVE – OR register with accumulator)  $[A] \leftarrow [A] \vee [r]$
- viii. **XRA M.** (EXCLUSIVE-OR memory with accumulator)  $[A] \leftarrow [A] \vee [[H-L]]$
- ix. **XRI data.** (EXCLUSIVE-OR immediate data with accumulator)  $[A] \leftarrow [A] \vee \text{data}$ .
- x. **CMA.** (Complement the accumulator)  $[A] \leftarrow [A]$
- x. **CMC.** (Complement the carry status)  $[CS] \leftarrow [CS]$
- xi. **STC.** (Set carry status)  $[CS] \leftarrow 1$ .
- xii. **CMP r.** (Compare register with accumulator)  $[A] - [r]$
- xiii. **CMP M.** (Compare memory with accumulator)  $[A] - [[H-L]]$
- xiv. **CPI data.** (Compare immediate data with accumulator)  $[A] - \text{data}$ .
- xv. **RLC** (Rotate accumulator left)  $[An+1] \leftarrow [An], [A0] \leftarrow [A7], [CS] \leftarrow [A7]$ .
  - The content of the accumulator is rotated left by one bit.
  - The seventh bit of the accumulator is moved to carry bit as well as to the zero bit of the accumulator.
  - Only CS flag is affected
- RRC.** (Rotate accumulator right)  $[A7] \leftarrow [A0], [CS] \leftarrow [A0], [An] \leftarrow [An+1]$ .
  - The content of the accumulator is rotated right by one bit.
  - The zero bit of the accumulator is moved to the seventh bit as well as to carry bit.
  - Only CS flag is affected.
- xvi. **RAL.** (Rotate accumulator left through carry)  $[An+1] \leftarrow [An], [CS] \leftarrow [A7], [A0] \leftarrow [CS]$ .
- xvii. **RAR.** (Rotate accumulator right through carry)  $[An] \leftarrow [An+1], [CS] \leftarrow [A0], [A7] \leftarrow [CS]$

### **Branch Group**

- The branching instruction alters the normal sequential flow.
- These instructions alter either unconditionally or conditionally

The instructions for conditional and unconditional jump, subroutine call and return, and restart.

Examples are: JMP, JC, JZ, CALL, CZ, RST etc.

- i. **JMP** addr (label). (Unconditional jump: jump to the instruction specified by the address).  $[PC] \leftarrow \text{Label}$ .
- ii. Conditional Jump addr (label): After the execution of the conditional jump instruction the program jumps to the instruction specified by the address (label) if the specified condition is fulfilled.
- iii. Specified condition is not fulfilled. If the condition is true and program jumps to the specified
  - a. **JZ** addr (label). (Jump if the result is zero)
  - b. **JNZ** addr (label) (Jump if the result is not zero)
  - c. **JC** addr (label). (Jump if there is a carry)
  - d. **JNC** addr (label). (Jump if there is no carry)
  - e. **JP** addr (label). (Jump if the result is plus)
  - f. **JM** addr (label). (Jump if the result is minus)
  - g. **JPE** addr (label) (Jump if even parity)
  - h. **JPO** addr (label) (Jump if odd parity)
- iv. **CALL** addr (label) (Unconditional CALL: call the subroutine identified by the operand)
  - CALL instruction is used to call a subroutine.
  - Before the control is transferred to the subroutine, the address of the next instruction of the main program is saved in the stack.
  - The content of the stack pointer is decremented by two to indicate the new stack top.
  - Then the program jumps to subroutine starting at address specified by the label.
- v. **RET** (Return from subroutine)
- vi. **RST n** (Restart) Restart is a one-word CALL instruction.
  - The content of the program counter is saved in the stack.
  - The program jumps to the instruction starting at restart location.

**I/O and Machine Control Group:**

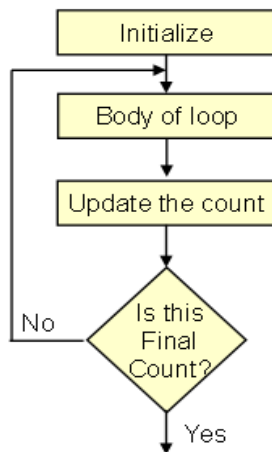
- Instructions for input/output ports stack and machine control.
- Examples are: IN, OUT, PUSH, POP, and HLT etc.
  - i. **IN** port-address. (Input to accumulator from I/O port)  $[A] \leftarrow [\text{Port}]$
  - ii. **OUT** port-address (Output from accumulator to I/O port)  $[\text{Port}] \leftarrow [A]$
  - iii. **PUSH** rp (Push the content of register pair to stack)
  - iv. **PUSH PSW** (PUSH Processor Status Word)
  - v. **POP** rp (Pop the content of register pair, which was saved, from the stack)



- vi. POP PSW (Pop Processor Status Word)
- vii. HLT (Halt)
- viii. XTHL (Exchange stack-top with H-L)
- ix. SPHL (Move the contents of H-L pair to stack pointer)
- x. EI (Enable Interrupts)
- xi. DI (Disable Interrupts)
- xii. SIM (Set Interrupt Masks)
- xiii. RIM (Read Interrupt Masks)
- xiv. NOP (No Operation)

**Counters**

- A loop counter is set up by loading a register with a certain value
- Then using the DCR (to decrement) and INR (to increment) the contents of the register are updated.
- A loop is set up with a conditional jump instruction that loops back or not depending on whether the count has reached the termination count.



Sample ALP for implementing a loop Using DCR instruction

```

    MVI C, 15H
    LOOP      DCR C
             JNZ LOOP
  
```

**Using a Register Pair as a Loop Counter**

Using a single register, one can repeat a Loop for a maximum count of 255 times

```

    LXI B, 1000H
LOOP DCX B
    MOV A, C
    ORA B
    JNZ LOOP

```

**Delays**

Delay = No. of T - States / Frequency

For example a “MVI” instruction uses 7 T - States.

If the Microprocessor is running at 2 MHz, the instruction would require 3.5 μSeconds to complete

**Delay loops**

Loops can be used to produce a certain amount of time delay in a program

**Example of a delay loop:**

```

    MVI C, FFH          7 T-States
LOOP DCR C             4 T-States
    JNZ LOOP          10 T-States

```

The first instruction initializes the loop counter and is executed only once requiring only 7 T-States.

The following two instructions form a loop that requires 14 T - States to execute and is repeated 255 times until C becomes 0

$$T_{\text{delay}} = T_O + T_L$$

$T_{\text{delay}}$  = total delay

$T_O$  = delay outside the loop

$T_L$  = delay of the loop

$T_O$  is the sum of all delays outside the loop.

Using these formulas the time delay can be calculated

$$T_O = 7 \text{ T- States}$$

**Delay of the MVI instruction**

$$T_L = (14 \times 255) - 3 = 3567 \text{ T-States}$$

14 T-States for the 2 instructions repeated 255 times ( $FF_{16} = 255_{10}$ ) reduced by the 3 T-States for the final JNZ

**Using a Register Pair as a Loop Counter**

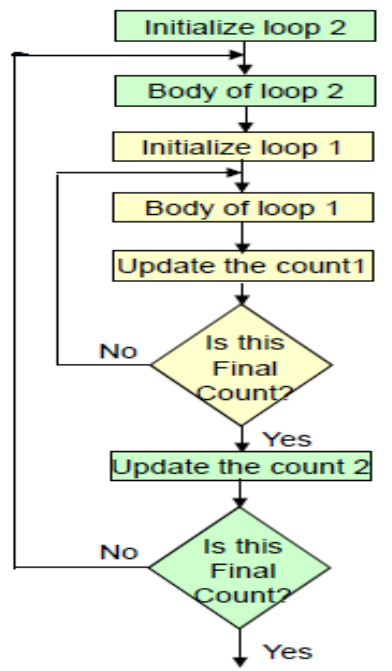
Using a single register, a loop for a maximum count of 255 times repeated

It is possible to increase this count by using a registerpair for the loop counter instead of the single register.

LXI B, 1000H	10 T-States
LOOP DCX B	6 T-States
MOV A, C	4 T-States
ORA B	4 T-States
JNZ LOOP	10 T-States

**Nested Loops**

Nested loops can be easily setup in Assembly language by using two registers for the two loop counters and updating the right register in the right loop.



**Stack operation in 8085 microprocessor.**

The stack is an area of memory identified by the programmer for temporary storage of information.

- The stack is a LIFO structure.
  - Last In First Out.
- The stack normally grows backwards into memory.
  - In other words, the programmer defines the bottom of the stack and the stack grows up into reducing address range.
- In the 8085, the stack is defined by setting the SP (Stack Pointer) register.
  - LXI SP, FFFFH
- This sets the Stack Pointer to location FFFFH (end of memory for the 8085).  
The SP pointer always points to “the top of the stack”.
- The Size of the stack is limited only by the available memory

**Saving Information on the Stack**

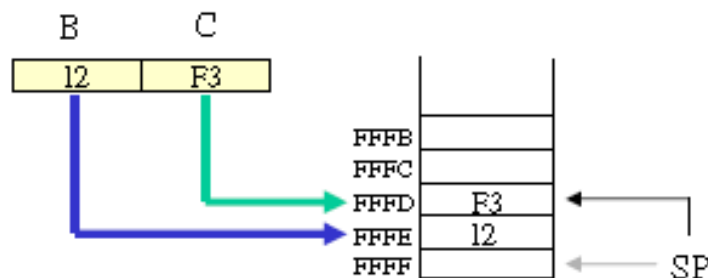
- Information is saved on the stack by PUSHing it on.
- It is retrieved from the stack by POPing it off.
- The 8085 provides two instructions: PUSH and POP for storing information on the stack and retrieving it back.
- Both PUSH and POP work with register pairs ONLY.

**On PUSH Instruction**

The stack pointer is decremented first, and then the information is placed on the stack.

**PUSH B (1 Byte Instruction)**

- Decrement SP
- Copy the contents of register B to the memory location pointed to by SP
- Decrement SP
- Copy the contents of register C to the memory location pointed to by SP



**The POP Instruction**

- Retrieving information from the stack
- The information is retrieved from the top of the the stack and then the pointer is incremented.

**POP B**

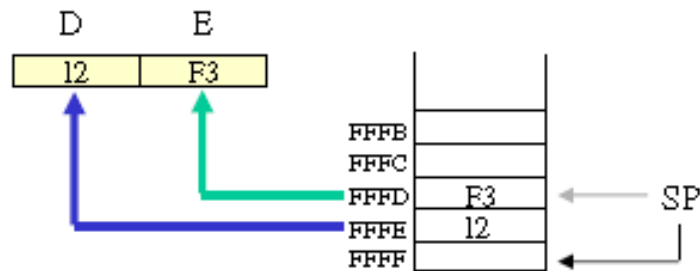
**POP D**

**POP H**

**POP PSW**

**POP D (1 Byte Instruction)**

- Copy the contents of the memory location pointed to by the SP to register E
- Increment SP
- Copy the contents of the memory location pointed to by the SP to register D
- Increment SP

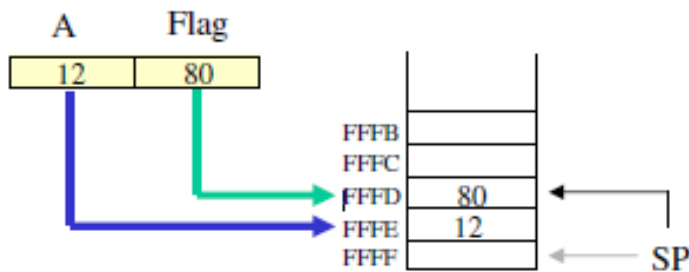


**The PSW Register Pair**

PSW register pair is made up of the Accumulator and the Flags register.

**PUSH PSW (1 Byte Instruction)**

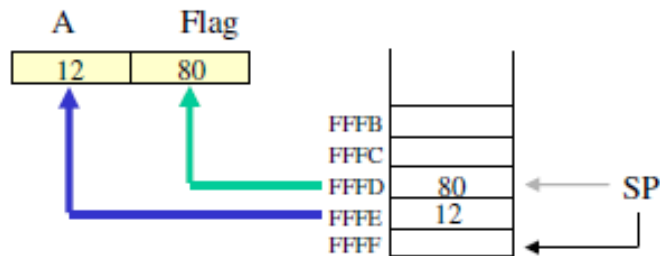
- Decrement SP
- Copy the contents of register A to the memory location pointed to by SP
- Decrement SP
- Copy the contents of Flag register to the memory location pointed to by SP



**POP PSW (1 Byte Instruction)**

- Copy the contents of the memory location pointed to by the SP to Flag register
- Increment SP
- Copy the contents of the memory location pointed to by the SP to register A

- Increment SP



### SPHL

- Copy H-L pair to the Stack Pointer (SP)
- Loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address.
- The contents of the H and L registers are not altered.

### XTHL

- The contents of L register are exchanged with the location pointed out by the contents of the SP.
- The contents of H register are exchanged with the next location (SP + 1).

### Subroutine in 8085 microprocessor.

- A subroutine is a group of instructions that will be used repeatedly in different locations of the program.
- Rather than repeat the same instructions several times, they can be grouped into a subroutine that is called from the different locations.
- In Assembly language, a subroutine can exist anywhere in the code.
- Subroutines must be placed separately from the main program.

The 8085 has two instructions for dealing with subroutines.

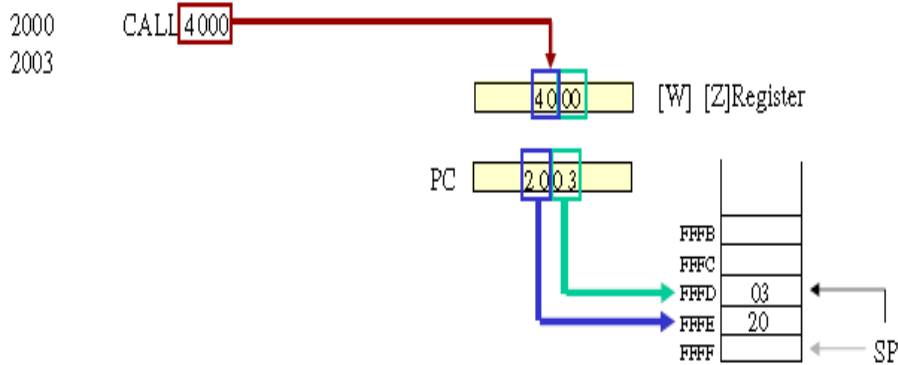
1. CALL instruction is used to redirect program execution to the subroutine.
2. RET instruction is used to return the execution to the calling routine

### **CALL Instruction**

#### **CALL 4000H (3 byte instruction)**

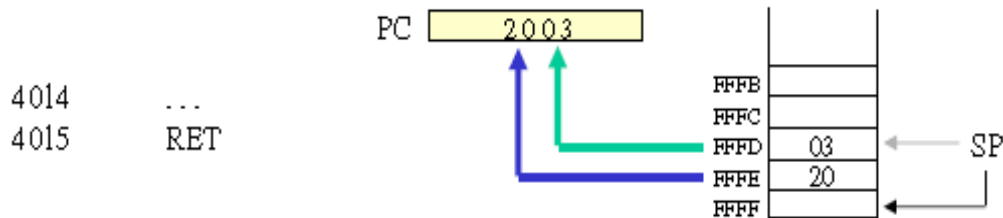
- When CALL instruction is fetched, the MP knows that the next two memory locations contain 16bit subroutine address in the memory.
- MP Reads the subroutine address from the next two memory location
- stores the higher order 8bit of the address in the W register

- stores the lower order 8bit of the address in the Z register
- Push the address of the instruction immediately following the CALL onto the stack [Return address]
- Loads the program counter with the 16-bit address supplied with the CALL instruction from WZ register.



**RET Instruction**

- RET ( 1 byte instruction)
  - Retrieve the return address from the top of the stack
  - Load the program counter with the return address.



**CALL conditionally**

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW.

Opcode	Description	Flag Status
CC	Call on Carry	CY = 1
CNC	Call on no Carry	CY = 0
CP	Call on positive	S = 0
CM	Call on minus	S = 1
CZ	Call on zero	Z = 1
CNZ	Call on no zero	Z = 0
CPE	Call on parity even	P = 1
CPO	Call on parity odd	P = 0

**RET conditionally**

- The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW.

Opcode	Description	Flag Status
RC	Return on Carry	CY = 1
RNC	Return on no Carry	CY = 0
RP	Return on positive	S = 0
RM	Return on minus	S = 1
RZ	Return on zero	Z = 1
RNZ	Return on no zero	Z = 0
RPE	Return on parity even	P = 1
RPO	Return on parity odd	P = 0

---



## 8085 PROGRAMS

### Addition of 2 – 8 bit Numbers

MVI	C, 00	Initialize C register to 00
LDA	4150	Load the value to Accumulator.
MOV	B, A	Move the content of Accumulator to B register.
LDA	4151	Load the value to Accumulator.
ADD	B	Add the value of register B to A
JNC	LOOP	Jump on no carry.
INR	C	Increment value of register C
LOOP:	STA	4152 Store the value of Accumulator (SUM).
	MOV	A, C Move content of register C to Acc.
	STA	4153 Store the value of Accumulator (CARRY)
	HLT	Halt the program.

### Subtraction of 2 – 8 bit Numbers

#### PROGRAM:

MVI	C, 00	Initialize C to 00
LDA	4150	Load the value to Acc.
MOV	B, A	Move the content of Acc to B register.

	LDA	4151	Load the value to Acc.
	SUB	B	
	JNC	LOOP	Jump on no carry.
	CMA		Complement Accumulator contents.
	INR	A	Increment value in Accumulator.
	INR	C	Increment value in register C
LOOP:	STA	4152	Store the value of A-reg to memory address.
	MOV	A, C	Move contents of register C to Accumulator.
	STA	4153	Store the value of Accumulator memory address.
	HLT		Terminate the program.

### Multiplication of 2 - 8 bit Numbers

#### PROGRAM:

	MVI	D, 00	Initialize register D to 00
	MVI	A, 00	Initialize Accumulator content to 00
	LXI	H, 4150	
	MOV	B, M	Get the first number in B - reg
	INX	H	
	MOV	C, M	Get the second number in C- reg.
LOOP:	ADD	B	Add content of A - reg to register B.
	JNC	NEXT	Jump on no carry to NEXT.
	INR	D	Increment content of register D
NEXT:	DCR	C	Decrement content of register C.
	JNZ	LOOP	Jump on no zero to address
	STA	4152	Store the result in Memory
	MOV	A, D	
	STA	4153	Store the MSB of result in Memory
	HLT		Terminate the program.

### DIVISION OF TWO 8 BIT NUMBERS

**PROGRAM:**

	LXI	H, 4150	
	MOV	B, M	Get the dividend in B – reg.
	MVI	C, 00	Clear C – reg for qoutient
	INX	H	
	MOV	A, M	Get the divisor in A – reg.
NEXT:	CMP	B	Compare A - reg with register B.
	JC	LOOP	Jump on carry to LOOP
	SUB	B	Subtract A – reg from B- reg.
	INR	C	Increment content of register C.
	JMP	NEXT	Jump to NEXT
LOOP:	STA	4152	Store the remainder in Memory
	MOV	A, C	
	STA	4153	Store the quotient in memory
	HLT		Terminate the program.

**Largest Number in an array**

**PROGRAM:**

	LXI	H,4200	Set pointer for array
	MOV	B,M	Load the Count
	INX	H	
	MOV	A,M	Set 1 <sup>st</sup> element as largest data
	DCR	B	Decrement the count
LOOP:	INX	H	
	CMP	M	If A- reg > M go to AHEAD
	JNC	AHEAD	
	MOV	A,M	Set the new value as largest
AHEAD:	DCR	B	
	JNZ	LOOP	Repeat comparisons till count = 0
	STA	4300	Store the largest value at 4300
	HLT		

**BCD to HEX Code conversion**

**PROGRAM:**

	LXI	H,4150	
	MOV	A,M	Initialize memory pointer
	ADD	A	MSD X 2
	MOV	B,A	Store MSD X 2
	ADD	A	MSD X 4
	ADD	A	MSD X 8
	ADD	B	MSD X 10
	INX	H	Point to LSD
	ADD	M	Add to form HEX
	INX	H	
	MOV	M,A	Store the result
	HLT		

### **UNIT III PERIPHERAL INTERFACING FOR 8085**

Study of Architecture and programming of ICs: 8255 PPI, 8259 PIC, 8251  
USART, 8279 Key board display controller and 8253 Timer/ Counter – Interfacing  
with 8085 - A/D and D/A converter interfacing.

## 8255 Programmable Peripheral Interface

The 8255A is a general purpose programmable I/O device designed to transfer the data from I/O to interrupt I/O under certain conditions as required. It can be used with almost any microprocessor.

It consists of three 8-bit bidirectional I/O ports (24 I/O lines) which can be configured as per the requirement.

### Features

- Widely used programmable parallel I/O device.
- 40 pin IC
- 24 I/O pins that can be grouped in two 8 bit parallel ports:  
A and B, with the remaining eight bits as port C.
- The 8 bits of port C can be used as individual bits or be grouped in two 4-bit ports:  
C UPPER (CU) and C LOWER (CL)

### Block diagram of 8255

It has a 40 pins of 4 groups.

1. Data bus buffer
2. Read Write control logic
3. Group A and Group B controls
4. Port A, B and C

### Data bus buffer:

- This is a tristate bidirectional buffer used to interface the 8255 to system data bus.
- Data is transmitted or received by the buffer on execution of input or output instruction by the CPU.

### Read/Write control logic:

- Accepts control signals (RD , WR ) and also inputs from address bus
- Issues commands to individual group of control blocks ( Group A, Group B).
- Has six signal lines. Which are RD, WR, RESET, CS, A0 and A1

### RD [READ] :-

- Enables the read operation.
- When low, the microprocessor reads data from a selected I/O port of 8255

### WR [WRITE] :-

- Enables the write operation.

- When low the MPU (microprocessor) writes into a selected I/O port or the control register.

**RESET:-**

- active high signal,
- A logic high on this line clears the control word register and set all ports in the input mode.

**CS [CHIP SELECT] :-**

If low it enables the 8255 to respond to RD and WR signals.

**A1 – A0 :-**

- Address lines driven by the microprocessor.
- These address lines are used for selecting any one of the three ports or a control word.

CS	A <sub>1</sub>	A <sub>0</sub>	Selected
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control Word Register
1	X	x	8255 is not selected

**Group A and Group B controls:**

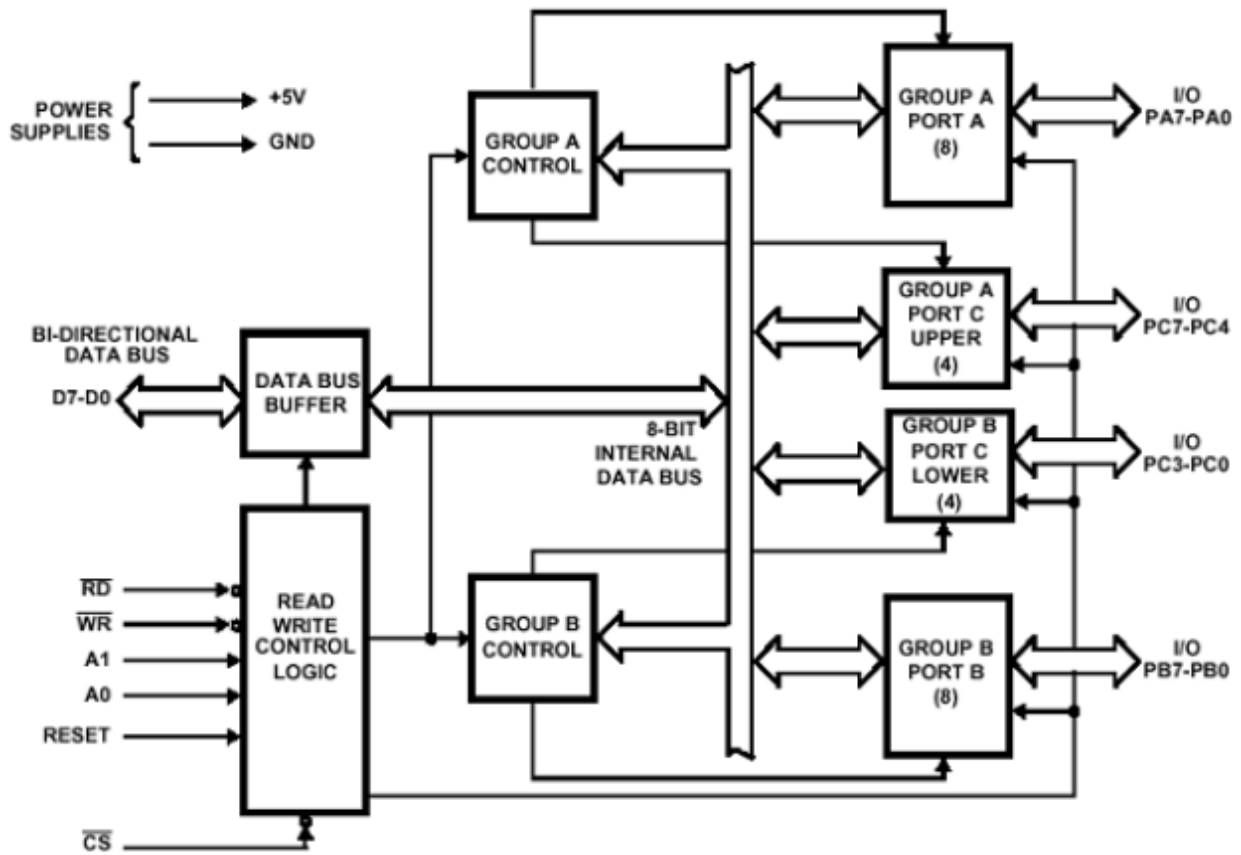
- Receive control from the CPU and issues commands to their respective ports.
- Group A - PA and PCU ( PC7 –PC4)
- Group B – PB and PCL ( PC3 – PC0)

*Ports of 8255A*

8255A has three ports, i.e., PORT A, PORT B, and PORT C.

- **Port A** contains one 8-bit output latch/buffer and one 8-bit input buffer.
- **Port B** is similar to PORT A.
- **Port C** can be split into two parts, i.e. PORT C lower (PC0-PC3) and PORT C upper (PC7-PC4) by the control word.

These three ports are further divided into two groups, i.e. Group A includes PORT A and upper PORT C. Group B includes PORT B and lower PORT C. These two groups can be programmed in three different modes, i.e. the first mode is named as mode 0, the second mode is named as Mode 1 and the third mode is named as Mode 2.



selection is done as follows:

D <sub>6</sub>	D <sub>5</sub>	Mode
0	0	0
0	1	1
1	X	2

As it is I/O mode, D<sub>7</sub> = 1.

For example, if port B and upper port C have to be initialized as input ports and lower port C and port A as output ports (all in mode 0):

1. Since it is an I/O mode, D<sub>7</sub> = 1.
2. Mode selection bits, D<sub>2</sub>, D<sub>5</sub>, D<sub>6</sub> are all 0 for mode 0 operation.
3. Port B and upper port C should operate as Input ports, hence, D<sub>1</sub> = D<sub>3</sub> = 1.
4. Port A and lower port C should operate as Output ports, hence, D<sub>4</sub> = D<sub>0</sub> = 0.

Hence, for the desired operation, the control word register will have to be loaded with "10001010" = 8A (hex).

**Mode 0 - simple I/O**

In this mode, the ports can be used for simple I/O operations without handshaking signals. Port A, port B provide simple I/O operation. The two halves of port C can be either used together as an additional



8-bit port, or they can be used as individual 4-bit ports. Since the two halves of port C are independent, they may be used such that one-half is initialized as an input port while the other half is initialized as an output port.

The input/output features in mode 0 are as follows:

1. Output ports are latched.
2. Input ports are buffered, not latched.
3. Ports do not have handshake or interrupt capability.
4. With 4 ports, 16 different combinations of I/O are possible.

'Latched' means the bits are put into a storage register (array of flip-flops) which holds its output constant even if the inputs change after being latched.

The 8255's outputs are latched to hold the last data written to them. This is required because the data only stays on the bus for one cycle. So, without latching, the outputs would become invalid as soon as the write cycle finishes.

The inputs are not latched because the CPU only has to read their current values, then store the data in a CPU register or memory if it needs to be referenced at a later time. If an input changes while the port is being read then the result may be indeterminate.

### **Mode 0 – input mode**

- In the input mode, the 8255 gets data from the external peripheral ports and the CPU reads the received data via its data bus.
- The CPU first selects the 8255 chip by making CS low. Then it selects the desired port using  $A_0$  and  $A_1$  lines.
- The CPU then issues an RD signal to read the data from the external peripheral device via the system data bus.

### **Mode 0 - output mode**

- In the output mode, the CPU sends data to 8255 via system data bus and then the external peripheral ports receive this data via 8255 port.

- CPU first selects the 8255 chip by making CS low. It then selects the desired port using A<sub>0</sub> and A<sub>1</sub> lines.
- CPU then issues a WR signal to write data to the selected port via the system data bus. This data is then received by the external peripheral device connected to the selected port.

### **Mode 1 - Strobed Input/output mode**

When we wish to use port A or port B for handshake (strobed) input or output operation, we initialise that port in mode 1 (port A and port B can be initialised to operate in different modes, i.e., for e.g., port A can operate in mode 0 and port B in mode 1). Some of the pins of port C function as handshake lines.

For port B in this mode (irrespective of whether is acting as an input port or output port), PC0, PC1 and PC2 pins function as handshake lines.

If port A is initialised as mode 1 input port, then, PC3, PC4 and PC5 function as handshake signals. Pins PC6 and PC7 are available for use as input/output lines.

The mode 1 which supports handshaking has following features:

1. Two ports i.e. port A and B can be used as 8-bit i/o ports.
2. Each port uses three lines of port c as handshake signal and remaining two signals can be used as i/o ports.
3. Interrupt logic is supported.
4. Input and Output data are latched.

### **Input Handshaking signals**

1. IBF (Input Buffer Full) - It is an output indicating that the input latch contains information.
2. STB (Strobed Input) - The strobe input loads data into the port latch, which holds the information until it is input to the microprocessor via the IN instruction.
3. INTR (Interrupt request) - It is an output that requests an interrupt. The INTR pin becomes a logic 1 when the STB input returns to a logic 1, and is cleared when the data are input from the port by the microprocessor.
4. INTE (Interrupt enable) - It is neither an input nor an output; it is an internal bit programmed via the port PC4(port A) or PC2(port B) bit position.

### **Output Handshaking signals**

1. OBF (Output Buffer Full) - It is an output that goes low whenever data are output(OUT) to the port A or port B latch. This signal is set to a logic 1 whenever the ACK pulse returns from the external device.

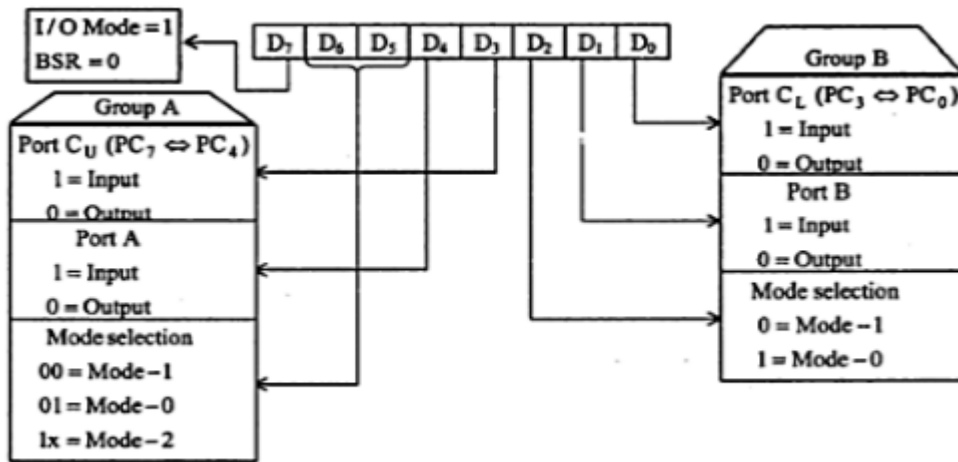
2. ACK (Acknowledge)-It causes the OBF pin to return to a logic 1 level. The ACK signal is a response from an external device, indicating that it has received the data from the 82C55A port.
3. INTR (Interrupt request) - It is a signal that often interrupts the microprocessor when the external device receives the data via the signal. this pin is qualified by the internal INTE(interrupt enable) bit.
4. INTE (Interrupt enable) - It is neither an input nor an output; it is an internal bit programmed to enable or disable the INTR pin. The INTE A bit is programmed using the PC6 bit and INTE B is programmed using the PC2 bit.

### **Mode 2 - Strobed Bidirectional Input/Output mode**

Only port A can be initialized in this mode. Port A can be used for *bidirectional handshake* data transfer. This means that data can be input or output on the same eight lines (PA0 - PA7). Pins PC3 - PC7 are used as handshake lines for port A. The remaining pins of port C (PC0 - PC2) can be used as input/output lines if group B is initialized in mode 0 or as handshaking for port B if group B is initialized in mode 1. In this mode, the 8255 may be used to extend the system bus to a slave [microprocessor](#) or to transfer data bytes to and from a [floppy disk](#) controller. Acknowledgement and handshaking signals are provided to maintain proper data flow and synchronisation between the data transmitter and receiver.

### **Control Word Register**

- contents of the control register called the control word
- Specifies an I/O function for each port.
- can only be written into and no read operation of the CW register is allowed.
- Bit D7 of the control register specifies either the I/O function or the Bit Set/Reset function.
- If Bit D7 =1, bits D6- D1 determine I/O functions in various mode
- If bit D7 = 0, port C operates in the Bit Set/Reset(BSR)mode



Control word with group definition

**MODES OF OPERATION OF 8255**

All the functions of 8255 A is classified according to two modes: the

- Bit Set/Reset (BSR) mode
- I/O mode.

**BSR mode**

- Concerned only with the 8 bit of port C
- Port C can be set or reset by writing an appropriate control word in the control register.
- A control word with bit D7=0 is recognized as a BSR control word.
- It does not alter any previously transmitted control word with bit d7=1 : thus the I/O operations of ports A & B are not affected by the BSR control word.
- In the BSR mode individual bits of port C can be used for applications such as an on/off switch.

**Control Word format in BSR mode**

D7	D6	D5	D4	D3	D2	D1	D0
0 (0=BSR)	X	X	X	B2	B1	B0	S/R (1=S,0=R)

Bit select: (Taking Don't care's as 0)

B2	B1	B0	PC bit	Control word (Set)	Control word (reset)
0	0	0	0	0000 0001 = 01h	0000 0000 = 00h
0	0	1	1	0000 0011 = 03h	0000 0010 = 02h
0	1	0	2	0000 0101 = 05h	0000 0100 = 04h
0	1	1	3	0000 0111 = 07h	0000 0110 = 06h
1	0	0	4	0000 1001 = 09h	0000 1000 = 08h
1	0	1	5	0000 1011 = 0Bh	0000 1010 = 0Ah
1	1	0	6	0000 1101 = 0Dh	0000 1100 = 0Ch
1	1	1	7	0000 1111 = 0Fh	0000 1110 = 0Eh

### **I/O MODE**

- Ports will be function as programmable I/O ports.  
Classified into three
- Mode 0: Simple Input or Output
- Mode 1: Input or Output with handshake
- Mode 2: Bidirectional Data Transfer

#### **Mode 0: Simple Input or Output**

- Basic I/O mode.
- ports A and B are used as two simple 8-bit I/O ports and port C as two 4-bit ports.
- Each port (or half-port in case of C) can be programmed to function as simply an input or an output port.

The input/output features in Mode 0 as follows:

1. Output is latched.
2. Inputs are not latched.
3. Ports do not have handshake or interrupt capability.
4. Any port can be used as input or output port.
5. 4-bit can combined used as a third 8-bit port.

#### **Mode 1: Input or Output with handshake**

- This is also called strobe I/O mode.
- Handshake signals are exchanged between the MPU and peripherals prior to data transfer.

The features of this mode include the following:

1. Port A and B function as 8-bit I/O ports. They can be configured either as input or output ports.
2. Each port uses three lines from port C as handshake signals. The remaining two lines of port C can be used for simple I/O functions.
3. Input and output data are latched.
4. Interrupt logic is supported.

The signals when port A & B are configured as output ports as follows:

### **Mode 2: Bidirectional Data Transfer**

- Strobe bi-directional I/O mode.
- Port A can be configured as the bidirectional port
- port B either in Mode 0 or Mode 1.
- Port A uses five signals from port C as handshake signals for data transfer.
- Remaining three signals from port C can be used as simple I/O or as handshake for port B.
- This mode is used primarily in applications such as data transfer between two computers of floppy disk controller interface.

### **Programmable Interrupt Controller(8259)**

#### **Features**

- 8 levels of interrupts.
- Can be cascaded in master-slave configuration to handle 64 levels of interrupts.
- Internal priority resolver.
- Fixed priority mode and rotating priority mode.
- Individually maskable interrupts.
- Modes and masks can be changed dynamically.
- Accepts IRQ, determines priority, checks whether incoming priority > current level being serviced, issues interrupt signal.
- In 8085 mode, provides 3 byte CALL instruction. In 8086 mode, provides 8 bit vector
- number.

- Polled and vectored mode.
- Starting address of ISR or vector number is programmable.
- No clock required

It includes 8 blocks

- Control Logic
- Read/Write logic
- Data bus buffer
- Three registers (IRR,ISR,IMR)
- Priority Resolver and
- Cascade Buffer

### **CONTROL LOGIC**

- Has two pins INT as an output, & INTA as an input
- INT pin is connected to the interrupt pin of MPU.

Whenever a valid interrupt is asserted, this signal goes high.

- INTA is the interrupt acknowledge signal from MPU

### **READ/WRITE LOGIC**

- When the address line A0 is at logic 0, the controller is selected to write a command or read a status
- Chip Select logic and A0 determine the port address of the controller

### **DATABUS BUFFER**

-This 3-state, bidirectional 8-bit buffer is used to interface the 8259 to the System Data Bus.  
Control words and status information are transferred through the Data Bus Buffer.

### **INTERRUPT REGISTERS & PRIORITY RESOLVER**

#### **Interrupt Request Register (IRR)**

8 input lines (IR0 –IR7) for interrupt

- When these lines go high, the requests are stored in the register.

#### **In-Service Register (ISR)**

Stores all the levels that are currently being serviced

#### **Interrupt Mask Register (IMR)**

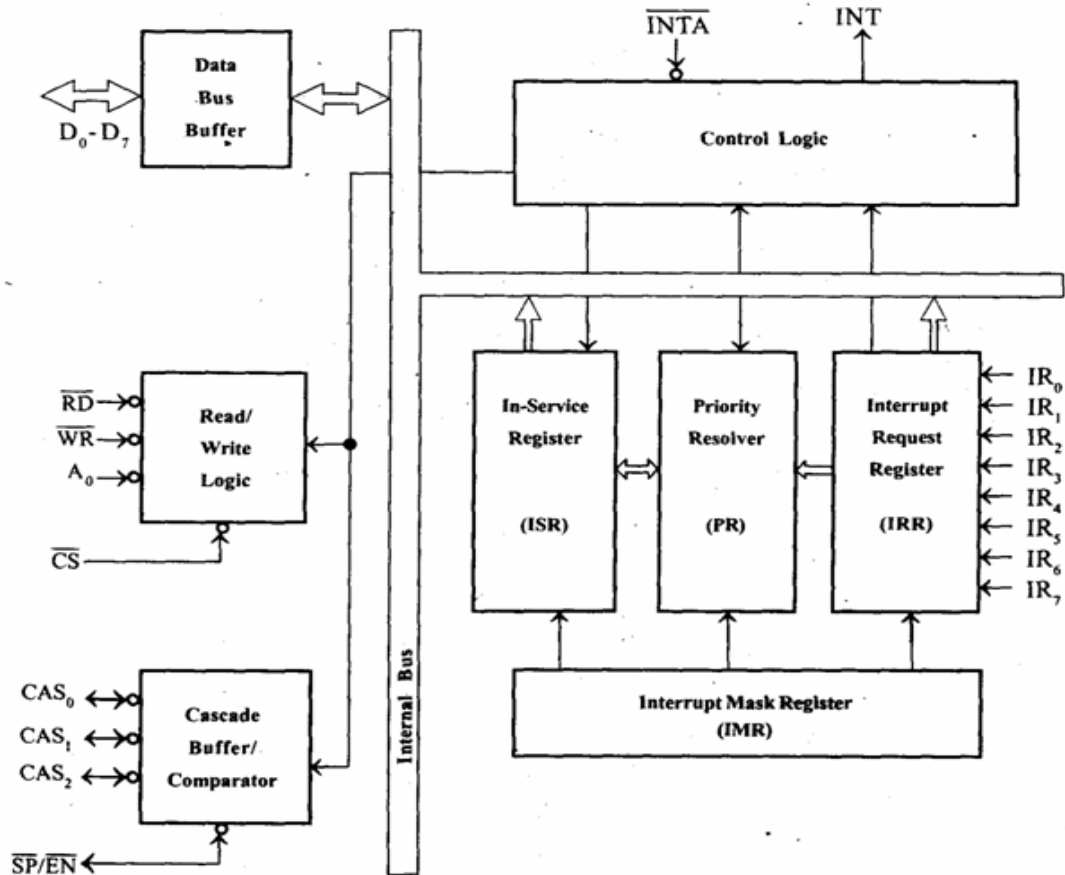
stores the masking bits of the interrupt lines to be masked.

#### **Priority Resolver(PR)**

Examines these 3 registers and determine whether INT should be sent to MPU

### **CASCADE BUFFER/COMPARATOR**

- Expands the number of interrupt levels by cascading two or more 8259



Internal Block Diagram of 8259

### INTERRUPT OPERATION

#### (Sequence of Events)

- The IRR stores the requests
- The priority resolver checks 3 registers:
  - IRR for interrupt requests
  - IMR for masking bits
  - ISR for serving the interrupt request
- It resolves the priority & sets the INT high when appropriate
  - o MPU acknowledges the interrupt by sending INTA
  - o Program sequence is transferred to the memory location specified by CALL instruction



**Modes of operation of 8259**

**PRIORITY MODES**

**Fully Nested Mode:**

- All IRs are arranged from highest to lowest (IR0 – highest & IR7 – lowest)
- In addition any IR can be assigned highest priority & priority sequence will begin at that IR.
- Ex. **IR4 has assigned highest priority**

<b>IR0</b>	<b>IR1</b>	<b>IR2</b>	<b>IR3</b>	<b>IR4</b>	<b>IR5</b>	<b>IR6</b>	<b>IR7</b>
<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>

**Automatic rotation mode**

- A device after being serviced, receives the lowest priority
- Ex. Assuming that IR2 has just been services, it will receive the seventh priority as shown

<b>IR0</b>	<b>IR1</b>	<b>IR2</b>	<b>IR3</b>	<b>IR4</b>	<b>IR5</b>	<b>IR6</b>	<b>IR7</b>
<b>5</b>	<b>6</b>	<b>7</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>

**Specific rotation mode**

- This mode is similar to the automatic rotation mode, except that the user can select any IR for the lowest priority, thus fixing all other priorities.

**END OF INTERRUPT**

**Three formats**

**- Nonspecific EOI Command:**

When this command is sent to the 8259A, it resets the highest priority ISR bit

**- Specific EOI Command:**

This command specifies which ISR bit to reset.

**- Automatic EOI:**

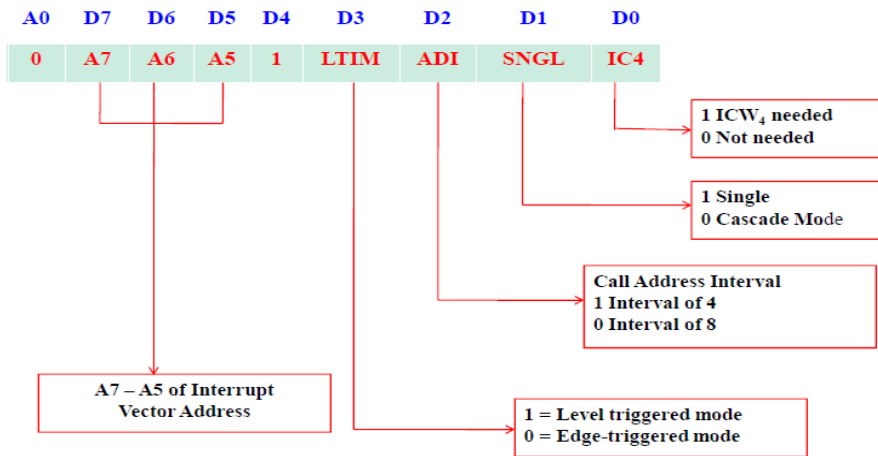
In this mode no command is necessary. During 3rd INTA the ISR bit is reset.

**PROGRAMMING THE 8259**

2 types of command words:

- Initialization command words
- Operational command words

**Initialization command words (ICW1)**



– The ICW1 specifies

- Single or multiple 8259As in the system
- 4 or 8 bit interval between the interrupt vector locations
- Address bits A7 – A5 of the CALL location; the rest are supplied by 8259A

A7    A6    A5    A4    A3    A2    A1    A0  
 0    1    1    0    0    0    0    0    = 60 H

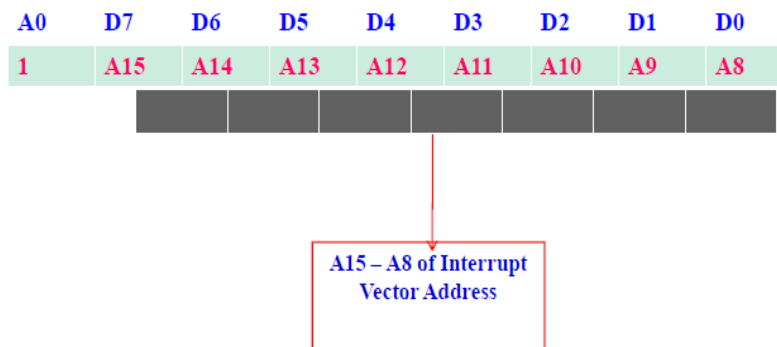
- The address bits A4 – A0 are supplied by the 8259A.

The subsequent addresses are four locations apart

- The port address of the 8259A for ICW1 is 80H;

A0 should be at logic 0, & the other bits are determined by the decoder.

**Initialization command words (ICW2)**



- Command word ICW2 is 20H, which specifies the high order byte of the Call address
- The port address of ICW2 is 81H; A0 should be at logic 1

**8253 programmable Timer / counter IC**

- is functionally similar to software designed counters and timers

- Generates accurate time delays can be used for applications  
real time clock, event counter, digital one-shot, square wave generator & a complex waveform generator

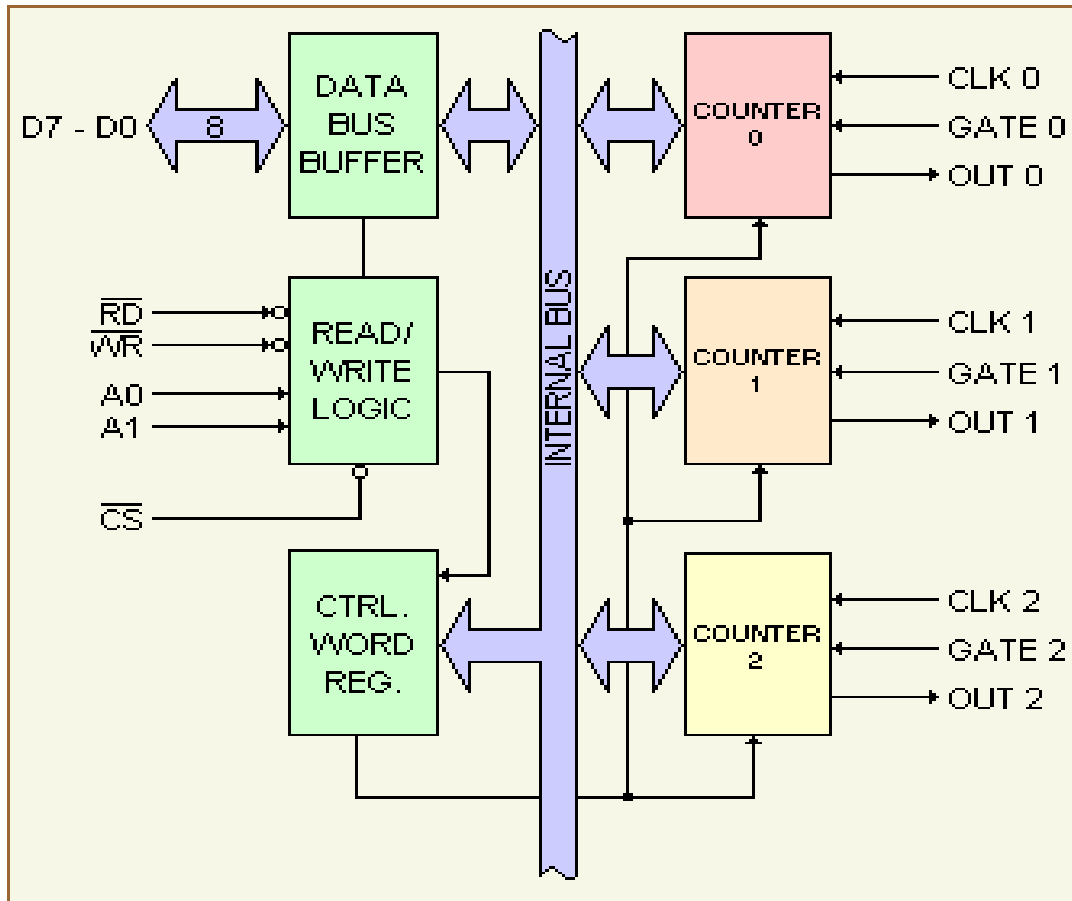
- Includes 3 identical counters that can operate in any one of 6 modes

**Operation**

A 16 bit count is loaded in its register and, on command, begins to decrement the count until it reaches 0

At the end of count it generates a pulse that can be used to interrupt the MPU

- The counter can count either in binary or BCD.



**BLOCK DIAGRAM OF 8254**

Three counters (0, 1 and 2)

- Data Bus buffer
- Read/Write control logic
- Control Register

Each counter has two input signals – clock (CLK) and GATE and one output signal – OUT.

**DATA BUS BUFFER:**

- 3-state, bi-directional, 8-bit buffer

- is used to interface the 8254 to the system bus.

**READ/WRITE CONTROL LOGIC :**

- Accepts inputs from the system bus
- Generates control signals for the other functional blocks of the 8254.
- A1 and A0 select one of the three counters or the Control

**The control section has five signals:**

RD(Read), WR(Write), CS(Chip Select), and address lines A1 and A0

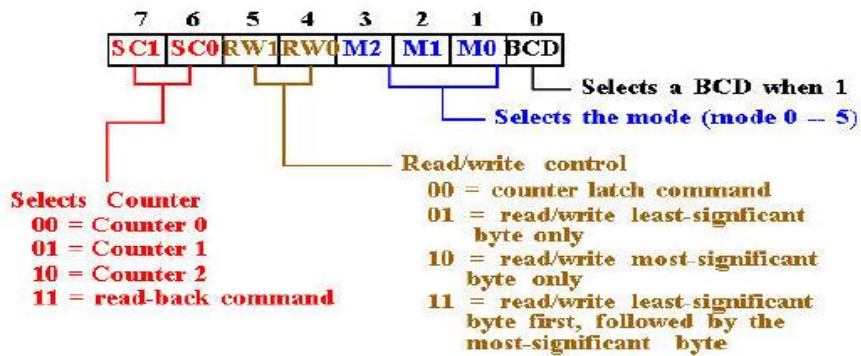
- Address lines A1 and A0 of the MPU are usually connected to A1 and A0 lines of the 8254,
- CS is tied to a decoded address

Control word register and counters are selected according to the signals on lines A1 and A0 as shown

A1	A0	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Register

**Control Word Register**

- Accessed when lines A1 and A0 are at logic 1.
- used to write a command word which specifies
  1. Counter to be used
  2. Mode
  3. Read or Write operation



Gate Settings of a Counter

<b>Signal Status Modes</b>	<b>Low Or Going Low</b>	<b>Rising</b>	<b>High</b>
0	Disables counting	—	Enables counting
1	—	1) Initiates counting 2) Resets output after next clock	—
2	1) Disables counting 2) Sets output immediately high	1) Reloads counter 2) Initiates counting	Enables counting
3	1) Disables counting 2) Sets output immediately high	1) Reloads counter 2) Initiates counting	Enables counting
4	Disables counting	—	Enables counting
5	—	Initiates counting	—

**Modes of Operation**

MODE 0: Interrupt on Terminal Count

MODE 1: Hardware – Retriggerable One-Shot

MODE 2 : Rate Generator

MODE 3: Square-Wave Generator

MODE 4: Software Triggered Strobe

MODE 5: Hardware Triggered Strobe

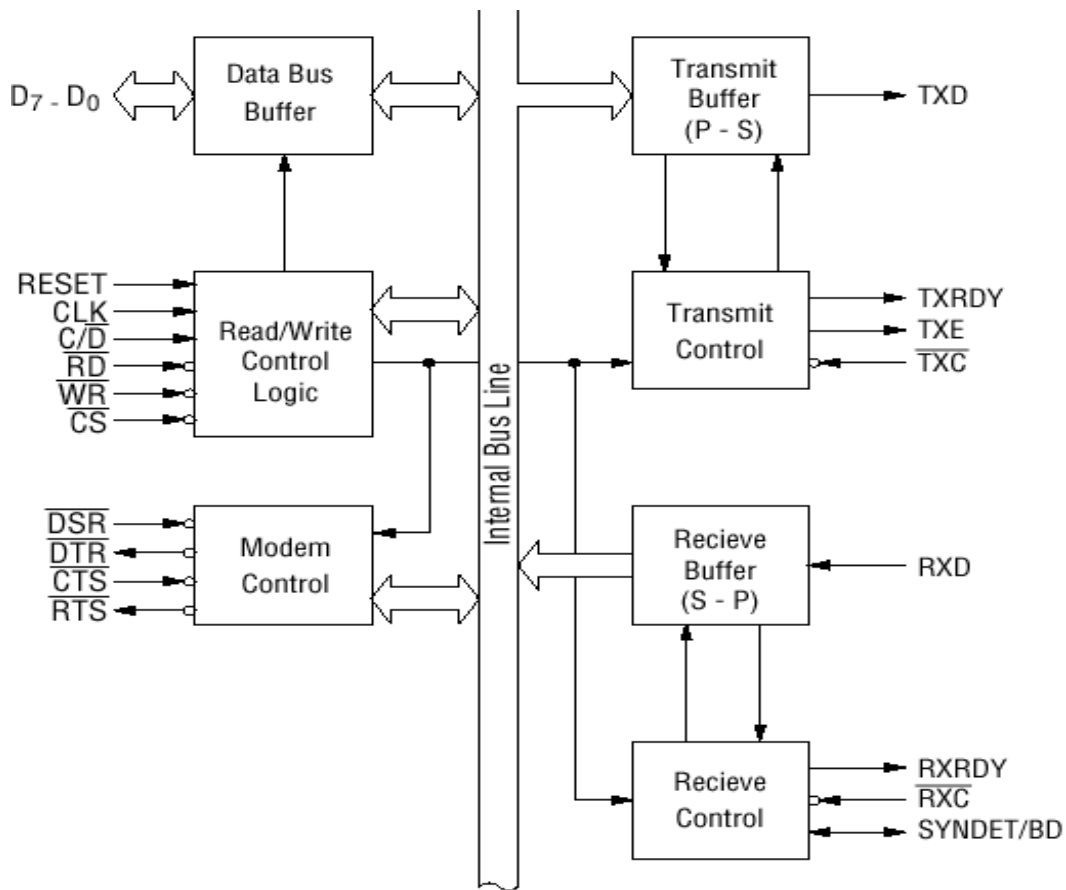
**8251 Universal Synchronous Asynchronous Receiver Transmitter (USART)**

The 8251 is a USART (Universal Synchronous Asynchronous Receiver Transmitter) for serial data communication.

- As a peripheral device of a microcomputer system, the 8251 receives parallel data from the CPU and transmits serial data after conversion. This device also receives serial data from the outside and transmits parallel data to the CPU after conversion.

$\overline{CS}$	$C/D$	$\overline{RD}$	$\overline{WR}$	
1	x	x	x	Data Bus 3-State
0	x	1	1	Data Bus 3-State
0	1	0	1	Status → CPU
0	1	1	0	Control Word ← CPU
0	0	0	1	Data → CPU
0	0	1	0	Data ← CPU

**Operation between a CPU and 8251**



**BLOCK DIAGRAM OF 8251 USART**

**Pin Description****D 0 to D 7 (I/O terminal)**

- Bidirectional data bus which receives control words and transmits data from the CPU and sends status words and received data to CPU.

**RESET (Input terminal)**

A "High" on this input forces the 8251 into "reset status." The device waits for the writing of "mode instruction." The min. reset width is six clock inputs during the operating status of CLK.

**CLK (Input terminal)**

CLK signal is used to generate internal device timing.

CLK signal is independent of RXC or TXC.

However, the frequency of CLK must be greater than 30 times the RXC and TXC at Synchronous mode and Asynchronous "x1" mode, and must be greater than 5 times at Asynchronous "x16" and "x64" mode.

**WR (Input terminal)**

This is the "active low" input terminal which receives a signal for writing transmit data and control words from the CPU into the 8251.

**RD (Input terminal)**

This is the "active low" input terminal which receives a signal for reading receive data and status words from the 8251.

**C/D (Input terminal)**

- This is an input terminal which receives a signal for selecting data or command words and status words when the 8251 is accessed by the CPU.
- If C/D = low, data will be accessed.
- If C/D = high, command word or status word will be accessed.

**CS (Input terminal)**

- This is the "active low" input terminal which selects the 8251 at low level when the CPU accesses.

**TXD (output terminal)**

- Output terminal for transmitting data from which serial-converted data is sent out.

**TXRDY (output terminal)**

- This is an output terminal which indicates that the 8251 is ready to accept a transmitted data character.

**TXEMPTY (Output terminal)**

- This is an output terminal which indicates that the 8251 has transmitted all the characters and had no data character.

**TXC (Input terminal)**

- This is a clock input signal which determines the transfer speed of transmitted data.
- In "synchronous mode," the baud rate will be the same as the frequency of TXC.
- In "asynchronous mode", it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16 or 1/64 the TXC. The falling edge of TXC sifts the serial data out of the 8251.

**RXD (input terminal)**

This is a terminal which receives serial data.

**RXRDY (Output terminal)**

This is a terminal which indicates that the 8251 contains a character that is ready to READ.

**RXC (Input terminal)**

- This is a clock input signal which determines the transfer speed of received data.
- In "synchronous mode," the baud rate is the same as the frequency of RXC.
- In "asynchronous mode," it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16, 1/64 the RXC.

**DSR (Input terminal)**

- This is an input port for MODEM interface.
- The input status of the terminal can be recognized by the CPU reading status words.

**DTR (Output terminal)**

- This is an output port for MODEM interface. It is possible to set the status of DTR by a command.

**CTS (Input terminal)**

This is an input terminal for MODEM interface which is used for controlling a transmit circuit.

**RTS (Output terminal)**

This is an output port for MODEM interface. It is possible to set the status RTS by a command

**Control Words**

There are two types of control word.

1. Mode instruction (setting of function)
2. Command (setting of operation)

**1) Mode Instruction**

Mode instruction is used for setting the function of the 8251.

- Writing of a control word after resetting will be recognized as a "mode instruction."



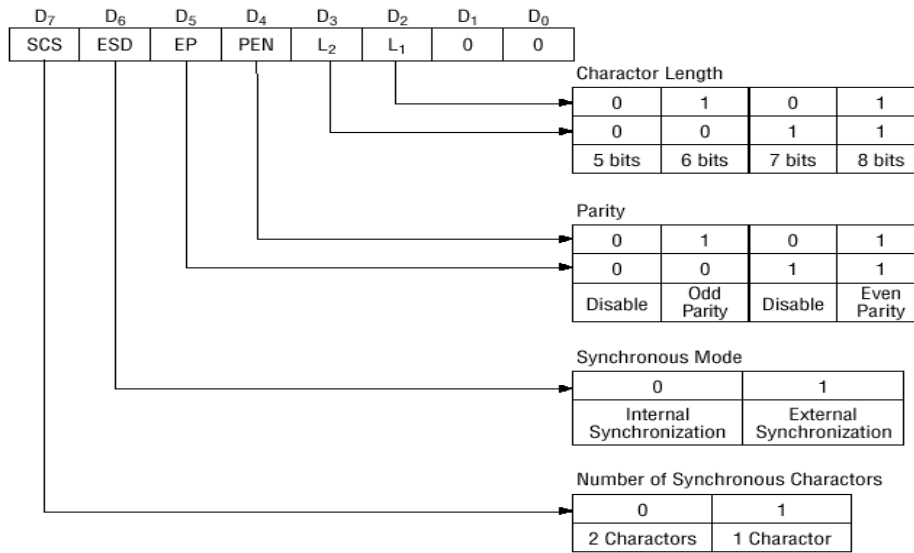


Fig. 3 Bit Configuration of Mode Instruction (Synchronous)

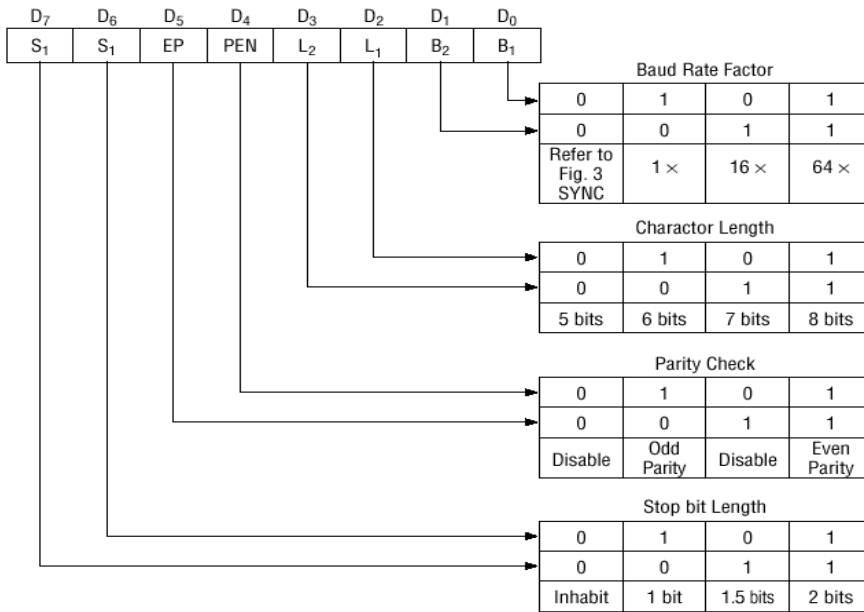


Fig. 2 Bit Configuration of Mode Instruction (Asynchronous)

2) Command

- Command is used for setting the operation of the 8251.
- It is possible to write a command whenever necessary after writing a mode instruction and sync characters.

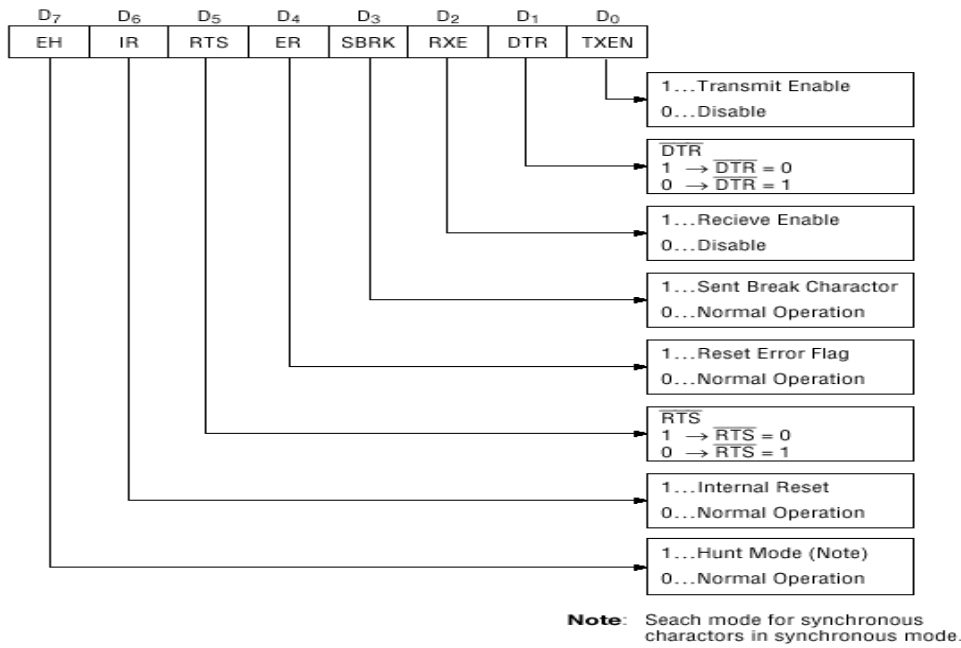


Fig. 4 Bit Configuration of Command

### Status Word

It is possible to see the internal status of the 8251 by reading a status word.

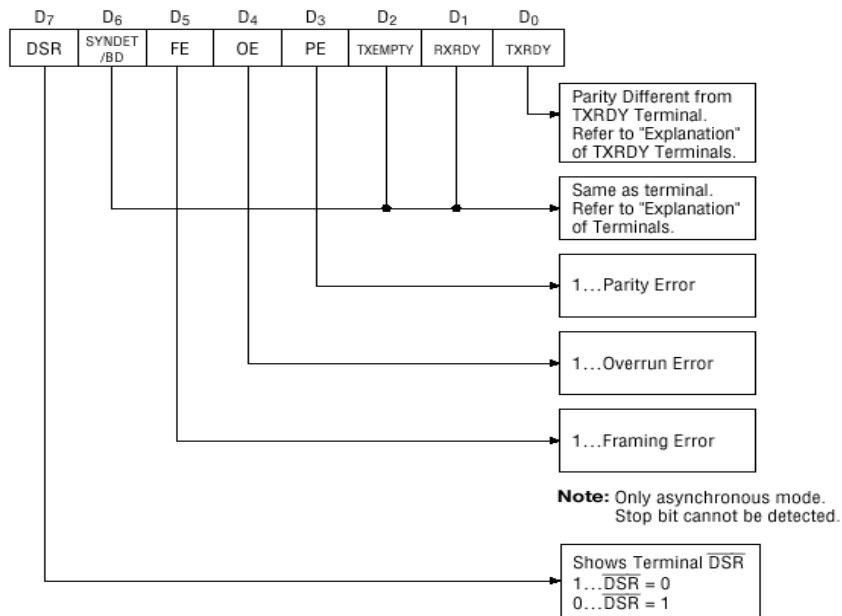


Fig. 5 Bit Configuration of Status Word

**8279 Programmable Keyboard / Display Controller**

8279 programmable keyboard/display controller is designed by Intel that interfaces a keyboard with the CPU. The keyboard first scans the keyboard and identifies if any key has been pressed. It then sends their relative response of the pressed key to the CPU and vice-a-versa.

**Features**

- Simultaneous keyboard and display operations.
- Scanned keyboard mode.
- Scanned sensor mode.
- 8-character keyboard FIFO.
- 1 6-character display.
- Right or left entry 1 6-byte display RAM.
- Programmable scan timing

**Sections**

1. Keyboard section
2. Display section
3. Scan Section
4. CPU interface section

**Keyboard section**

- Eight return lines RL0 - RL7 that can be used to form the columns of a keyboard matrix.
- It has two additional inputs: shift and control/strobe.
- Operating modes of keyboard section
  1. 2-key lockout
  2. N-key rollover.

**2-key lockout mode,**

- If two keys are pressed simultaneously, only the first key is recognized.

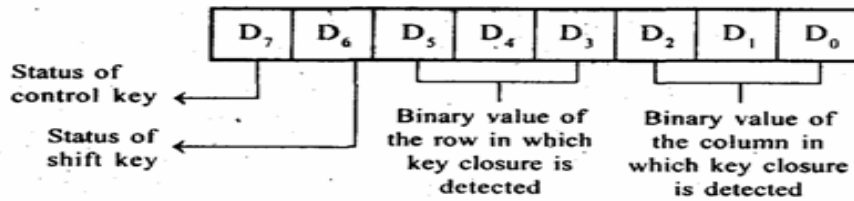
**N-key rollover mode**

- Simultaneous keys are recognized and their codes are stored in FIFO.

**8 x 8 FIFO (First In First Out) RAM.**

- The FIFO can store eight key codes in the scan keyboard mode.
- The status of the shift key and control key are also stored along with key code.
- The 8279 generate an interrupt signal when there is an entry in FIFO.

**The format of key code entry in FIFO for scan keyboard mode is,**



### Display section

- Eight output lines divided into two groups A0-A3 and B0-B3.
- The output lines can be used either as a single group of eight lines or as two groups of four lines, in conjunction with the scan lines for a multiplexed display.
- Output lines are connected to the anodes through driver transistor in case of common cathode 7-segment LEDs.
- The cathodes are connected to scan lines through driver transistors.
- The display can be blanked by BD (low) line.
- The display section consists of 16 x 8 display RAM. The CPU can read from or write into any location of the display RAM

### Scan section

- scan counter and four scan lines, SL0 to SL3

#### Decoded scan mode,

- Output of scan lines will be similar to a 2-to-4 decoder.

#### Encoded scan mode,

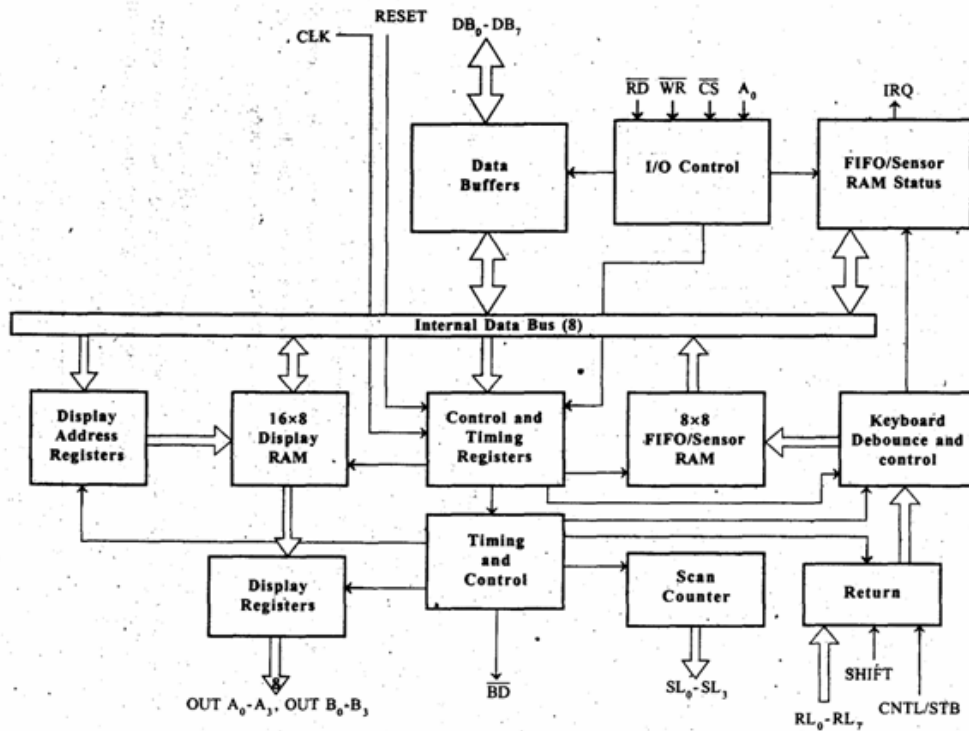
- Output of scan lines will be binary count, and so an external decoder should be used to convert the binary count to decoded output.
- The scan lines are common for keyboard and display.

The scan lines are used to form the rows of a matrix keyboard and also connected to digit drivers of a multiplexed display, to turn ON/OFF.

### CPU interface section

- Takes care of data transfer between 8279 and the processor.
- Eight bidirectional data lines DB0 to DB7 for data transfer between 8279 and CPU.
- two internal address A =0 for selecting data buffer and A = 1 for selecting control register of 8279.
- The control signals WR (low), RD (low), CS (low) and A0 are used for read/write to 8279.
- It has an interrupt request line IRQ, for interrupt driven data transfer with processor.
- The 8279 require an internal clock frequency of 100 kHz. This can be obtained by dividing the input clock by an internal prescaler.

- The RESET signal sets the 8279 in 16-character display with two -key lockout keyboard modes



**8279 Programmable Keyboard / Display Controller Functional Block Diagram**

**Keyboard Interface of 8279**

D7	D6	D5	Function	Purpose
0	0	0	Mode set	Selects the number of display positions, type of key scan...
0	0	1	Clock	Programs internal clk, sets scan and debounce times.
0	1	0	Read FIFO	Selects type of FIFO read and address of the read.
0	1	1	Read Display	Selects type of display read and address of the read.
1	0	0	Write Display	Selects type of write and the address of the write.
1	0	1	Display write inhibit	Allows half-bytes to be blanked.
1	1	0	Clear	Clears the display or FIFO
1	1	1	End interrupt	Clears the IRQ signal to the microprocessor.

**Keyboard Interface of 8279**

First three bits given below select one of 8 control registers (opcode).

0	0	0	D	D	K	K	K
---	---	---	---	---	---	---	---

Mode set: Opcode 000.  
 DD sets displays mode.  
 KKK sets keyboard mode.

DD field selects either:

- 8- or 16-digit display
- Whether new data are entered to the rightmost or leftmost display position.

DD	Function
00	8-digit display with left entry
01	16-digit display with left entry
10	8-digit display with right entry
11	16-digit display with right entry

**Keyboard Interface of 8279**

KKK	Function
000	Encoded keyboard with 2-key lockout
001	Decoded keyboard with 2-key lockout
010	Encoded keyboard with N-key rollover
011	Decoded keyboard with N-key rollover
100	Encoded sensor matrix
101	Decoded sensor matrix
110	Strobed keyboard, encoded display scan
111	Strobed keyboard, decoded display scan

Encoded: SI outputs are active-high, follow binary bit pattern 0-7 or 0-15.

Decoded: SL outputs are active-low (only one low at any time).

- Pattern output: 1110, 1101, 1011, 0111.

Strobed: An active high pulse on the CN/ST input pin strobes data from the RL pins into an internal FIFO for reading by micro later.

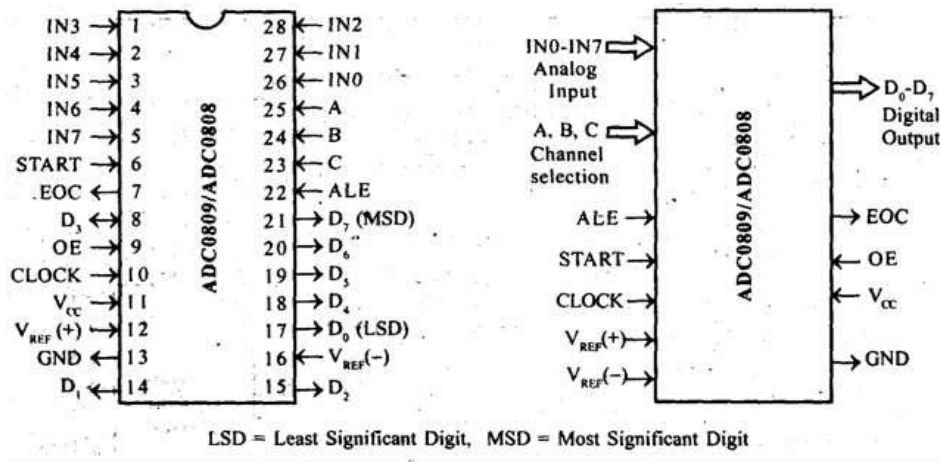
2-key lockout/N-key rollover: Prevents 2 keys from being recognized if pressed simultaneously/Accepts all keys pressed from 1st to last.

### ADC (IC 0809) Interfacing with 8085 Microprocessor

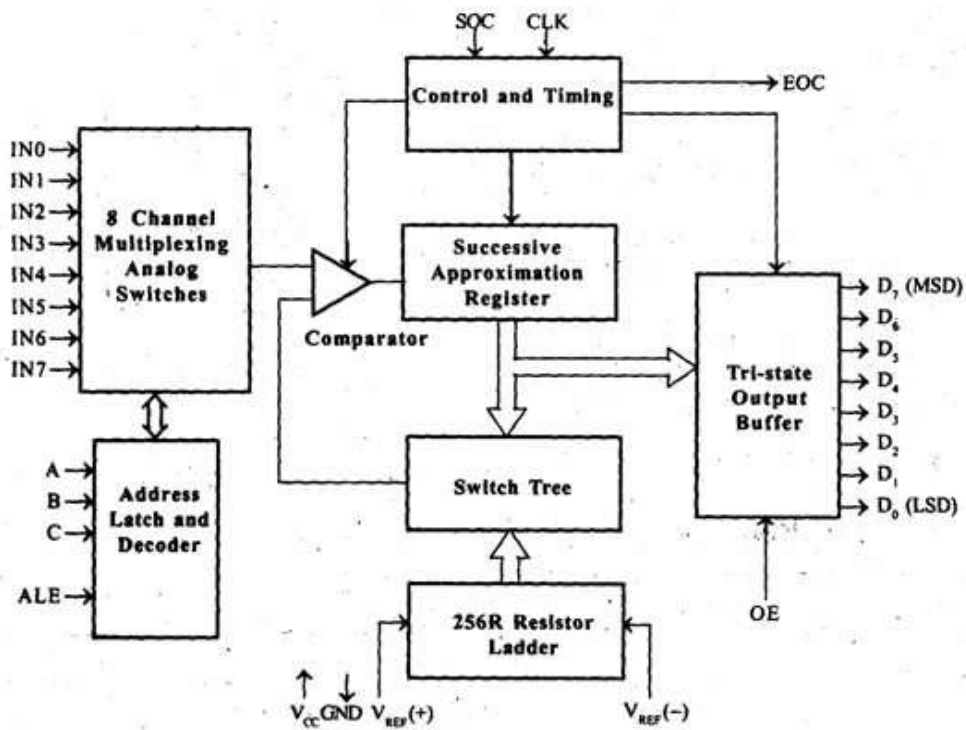
#### Features

- 8-bit successive approximation type ADC with inbuilt 8-channel multiplexer.
- suitable for interface with 8086 microprocessor.
- available as a 28 pin IC in DIP (Dual Inline Package).
- has a total unadjusted error of  $\pm 1$  LSD (Least Significant Digit).
- The ADC0808 is also same as ADC0809 except the error. The total unadjusted error in ADC0808 is  $\pm 1/2$  LSD.

#### Pin Diagram of ADC 0809



#### Block Diagram of ADC 0809



- The successive approximation register (SAR) performs eight iterations to determine the digital code for input value.
- The SAR is reset on the positive edge of START pulse and start the conversion process on the falling edge of START pulse.
- A conversion process will be interrupted on receipt of new START pulse.
- The End-Of-Conversion (EOC) will go low between 0 and 8 clock pulses after the positive edge of START pulse.
- The ADC can be used in continuous conversion mode by tying the EOC output to START input.
- In this mode an external START pulse should be applied whenever power is switched ON.
- The 256R ladder network has been provided instead of conventional R/2R ladder because of its inherent monotonic, which guarantees no missing digital codes. Also the 256R resistor network does not cause load variations on the reference voltage.
- The comparator in ADC0809/ADC0808 is a chopper- stabilized comparator.
- It converts the DC input signal into an AC signal, and amplifies the AC sign using high gain AC amplifier.
- Then it converts AC signal to DC signal.
- This technique limits the drift component of the amplifier, because the drift is a DC component and it is not amplified/passed by the AC amplifier.



- This makes the ADC extremely insensitive to temperature, long term drift and input offset errors. In ADC conversion process the input analog value is quantized and each quantized analog value will have a unique binary equivalent.
- The quantization step in ADC0809/ADC0808 is given by

$$Q_{\text{step}} = \frac{V_{\text{REF}}}{2^8} = \frac{V_{\text{REF}(+)} - V_{\text{REF}(-)}}{256_{10}}$$

### DAC (0800) Interfacing with 8085 Microprocessor

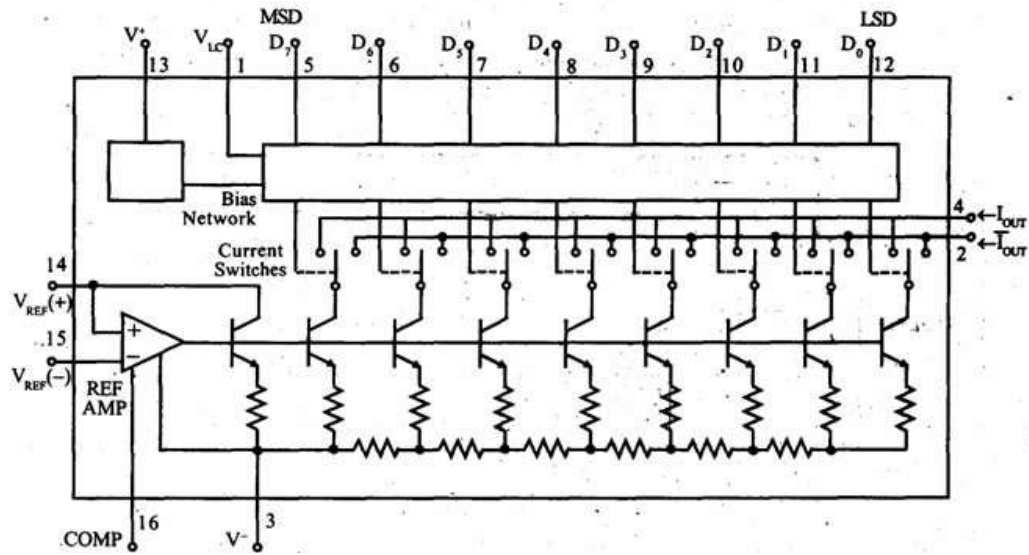
#### Features

- To convert the digital signal to analog signal a Digital-to-Analog Converter (DAC) has to be employed.
- The DAC will accept a digital (binary) input and convert to analog voltage or current.
- Every DAC will have "n" input lines and an analog output.
- The DAC require a reference analog voltage (Vref) or current (Iref) source.

#### Resolution.

- The smallest possible analog value that can be represented by the n-bit binary code.
- The resolution of DAC with n-bit binary input is  $1/2^n$  of reference analog value.

### Circuit Diagram of DAC 0800



- The DAC0800 is an 8-bit, high speed, current output DAC with a typical settling time (conversion time) of 100 ns.
- It produces complementary current output, which can be converted to voltage by using simple resistor load.
- The DAC0800 require a positive and a negative supply voltage in the range of ± 5V to ±18V.
- It can be directly interfaced with TTL, CMOS, PMOS and other logic families.
- For TTL input, the threshold pin should be tied to ground (V<sub>LC</sub> = 0V).
- The reference voltage and the digital input will decide the analog output current, which can be converted to a voltage by simply connecting a resistor to output terminal or by using an op-amp I to V converter.
- The DAC0800 is available as a 16-pin IC in DIP.

**ADC Conversion Table**

Digital Input	Analog Output
0000 0000	$\frac{0}{2^8} \times 5$ Volts
0000 0001	$\frac{1}{2^8} \times 5$ Volts
0000 0010	$\frac{2}{2^8} \times 5$ Volts
0000 0011	$\frac{3}{2^8} \times 5$ Volts
⋮	⋮
1111 1111	$\frac{255}{2^8} \times 5$ Volts

## UNIT – I

### 1. What is microprocessor?

A microprocessor is a multipurpose, programmable logic device that reads binary instructions from a storage device called memory accepts binary data as input and processes data according to those instructions and provide result as output.

### 2. What is the function of IO/M signal in the 8085?

It is a status signal. It is used to differentiate between memory locations and I/O operations.

When this signal is low ( $IO/M = 0$ ) it denotes the memory related operations.

When this signal is high ( $IO/M = 1$ ) it denotes an I/O operation.

### 3. Mention the purpose of SID and SOD lines.

SID (Serial input data line): It is an input line through which the microprocessor accepts serial data.

SOD (Serial output data line): It is an output line through which the microprocessor sends output serial data.

### 4. List the flags of 8085?

There are five flags in 8085.

They are sign flag, zero flag, auxiliary carry flag, parity flag and carry flag.

### 5. List the four instructions which control the interrupt structure of the 8085 microprocessor.

DI ( Disable Interrupts )

EI ( Enable Interrupts )

RIM ( Read Interrupt Masks )

SIM ( Set Interrupt Masks )

### 6. What is meant by interrupt?

Interrupt is an external signal that causes a microprocessor to jump to a specific subroutine.

**7. Explain priority interrupts of 8085.**

Interrupts	Priority
TRAP	1
RST 7.5	2
RST 6.5	3
RST 5.5	4
INTR	5

**8. What is the use of ALE?**

The ALE (Address latch enable) is a signal used to demultiplex the address and data lines using an external latch. It is used to enable the external latch.

**9. Define instruction cycle, machine cycle and T-state.**

**Instruction cycle**- defined as the time required to complete the execution of an instruction.

**Machine cycle** - defined as the time required to complete one operation of accessing memory, I/O or acknowledging an external request.

**T-cycles** - defined as one subdivision of the operation performed in one clock period.

**10. Distinguish between the memory mapped I/O peripheral I/O?**

SI: NO	Memory Mapped I/O	Peripheral I/O
1	16-bit device address	8-bit device address
2	Data transfer between any general purpose register and I/O port.	Data is transfer only between accumulator and I.O port
3	The memory map (64K) is shared between I/O device and system memory.	The I/O map is independent of the memory map;256 input device and 256 output device can be connected
4	More hardware is required to decode 16-bit address	Less hardware is required to decode 8-bit address
5	Arithmetic or logic operation can be directly performed with I/O data	Arithmetic or logical operation cannot be directly performed with I/O data

**11. What is the signal classification of 8085?**

All the signals of 8085 can be classified into 6 groups

1. Address bus
2. Data bus
3. Control and status signals
4. Power supply and frequency signals
5. Externally initiated signals
6. Serial I/O ports

**12. How many machine cycles does 8085 have, mention them**

The 8085 has seven machine cycles. They are

?? Opcode fetch

?? Memory read

?? Memory write

?? I/O read

?? I/O write

?? Interrupt acknowledge

?? Bus idle.

**13. What are the different types of interrupts?**

1. ? Hardware
2. Software

**Hardware interrupts-** The interrupts where the CPU pins are used to receive interrupt requests , are called hardware interrupts.

**Software interrupts –** This interrupt is caused by the execution of the instruction. These are special instructions supported by the microprocessor.

**14. What is vectored and non-vectored interrupt?**

**Vectored interrupt** - When an interrupt is accepted, if the processor control branches to a specific address defined by the manufacturer then the interrupt is called vectored interrupt.

**Non-vectored interrupt** there is no specific address for storing the interrupt service routine. Hence the interrupted device should give the address of the interrupt service routine.

**15. List the software and hardware interrupts of 8085?**

Software interrupts : RST 0,RST 1,RST 2,RST 3,RST 4,RST 5,RST 6,RST 7

Hardware interrupts : TRAP,RST 7.5,RST 6.5,RST 5.5, INTR.

**UNIT - II**

**1. What are the functions of an accumulator?**

The accumulator is the register associated with the ALU operations and sometimes I/O operations.

It is an integral part of ALU. It holds one of data to be processed by ALU.

It also temporarily stores the result of the operation performed by the ALU.

**2. List the allowed register pairs of 8085.**

B-C register pair

D-E register pair

H-L register pair

**3. List out the five categories of the 8085 instructions. Give examples of the instructions for each group.**

Data transfer group – MOV, MVI, LXI.

Arithmetic group – ADD, SUB, INR.

Logical group –ANA, XRA, CMP.

Branch group – JMP, JNZ, CALL.

Stack I/O and Machine control group – PUSH, POP, IN, HLT.

**4. Explain the difference between a JMP instruction and CALL instruction.**

A JMP instruction permanently changes the program counter.

A CALL instruction leaves information on the stack so that the original program execution sequence can be resumed.

**5. Explain the purpose of the I/O instructions IN and OUT.**

The IN instruction is used to move data from an I/O port into the accumulator.

The OUT instruction is used to move data from the accumulator to an I/O port.

The IN & OUT instructions are used only on microprocessor which use a separate address space for interfacing.

**6. What is the difference between the shift and rotate instructions?**

A rotate instruction is a closed loop instruction. That is, the data moved out at one end is put back in at the other end.

The shift instruction loses the data that is moved out of the last bit locations.

**7. What is an op-code field?**

The op-code field contains the instruction code (mnemonics). The assembler will translate these mnemonics into binary.

**8. What is an operand field?**

Operand field contain the data, register name (A, B, C etc.) or address of the memory location on which on which the operation is to be performed.

**9. Explain LDA, STA and DAA instructions**

LDA copies the data byte into accumulator from the memory location specified

by the 16-bit address.

STA copies the data byte from the accumulator in the memory location specified by 16-bit address.

DAA changes the contents of the accumulator from binary to 4-bit BCD digits.

**10. Explain the different instruction formats with examples**

The instruction set is grouped into the following formats

One byte instruction MOV C,A

Two byte instruction MVI A,39H

Three byte instruction JMP 2345H

**11. What is the use of addressing modes , mention the different types**

The various formats of specifying the operands are called addressing modes, it is used to access the operands or data. The different types are as follows

Immediate addressing

Register addressing

Direct addressing

Indirect addressing

Implicit addressing

**12. Define stack and explain stack related instructions**

The stack is a group of memory locations in the R/W memory that is used for the temporary storage of binary information during the execution of the program.

The stack related instructions are PUSH & POP

**13. Why do we use XRA A instruction**

The XRA A instruction is used to clear the contents of the Accumulator and store the value 00H.

**14. Compare RET and POP**



<b>RET</b>	<b>POP</b>
RET transfers the contents of the top two locations of the stack to the PC	POP transfers the contents of the top two locations of the stack to the specified register pair
When RET is executed the SP is incremented by two	When POP is executed the SP is incremented by two
Has 8 conditional RETURN instructions	No conditional POP instructions

**15. Write 8085 assembly language instructions to store the contents of the flag register in memory location 2000H.**

```
PUSH PSW
POP B
MOV A,C
STA 2000H.
HLT
```

**16. What is program counter? How is it useful in program execution?**

The program counter keeps track of program execution.

To execute a program the starting address of the program is loaded in program counter.

The PC sends out an address to fetch a byte of instruction from memory and increments its content automatically.

**17. What is subroutine?**

A subroutine is a group of instructions that performs a subtask of repeated occurrence.

**18. What are the two compare instructions available in 8085?**

CMP R : Compare the contents of register with the contents of A

CPI 8-bit: Compare 8-bit data with the contents of A

**19. List some of the Logical instructions in 8085.**

ANA R

ANI 8-bit

ANA M

ORA R

ORI 8-bit

ORA M

XRA R

XRI 8-bit

XRA M

**20. List some of the arithmetic operations.**

ADD, SUB, INR, DCR

**UNIT - III**

**1. List the operation modes of 8255**

a) I.O Mode

i.Mode 0-Simple Input/Output.

ii.Mode 1-Strobed Input/Output (Handshake mode)

iii.Mode 2-Strobed bi-directional mode

b) Bit Set/Reset Mode.

**2. What is a control word?**

It is a word stored in a register (control register) used to control the operation of a program digital device.

**3. What is the purpose of control word written to control register in 8255?**

The control words written to control register specify an I/O function for each I.O port. The bit D7 of the control word determines either the I/O function of the BSR function.

**4. What is the size of ports in 8255?**

Port-A : 8-bits

Port-A : 8-bits

Port-B : 8-bits

Port-CU : 4-bits

Port-CL : 4-bits

**5. What is an USART?**

USART stands for universal synchronous/Asynchronous Receiver/ Transmitter.

It is a programmable communication interface that can communicate by using either synchronous or asynchronous serial data.

**6. What are the functional types used in control words of 8251?**

The control words of 8251A are divided into two functional types.

**Mode Instruction control word**

This defines the general operational characteristics of 8251A.

**Command Instruction control word:-**

The command instruction controls the actual operations of the selected format like enable transmit/receive, error reset and modem control.

**7. List the major components of the keyboard/Display interface.**

Keyboard section

Scan section

Display section

CPU interface section.

### **8. What are the modes of operations used in 8253?**

Each of the three counters of 8253 can be operated in one of the following six modes of operation.

Mode 0 (Interrupt on terminal count)

Mode 1 (Programmable monoshot)

Mode 2 (Rate generator)

Mode 3 (Square wave generator)

Mode 4 (Software triggered strobe)

Mode 5 (Hardware triggered strobe)

### **9. Give different types of command words used in 8259A?**

The command words of 8259A are classified in two groups

Initialization command words (ICWs)

Operation command words (OCWs)

### **10. What is Key bouncing?**

Mechanical switches are used as keys in most of the keyboards. When a key is pressed the contact bounce back and forth and settle down only after a small time delay (about 20ms). Even though a key is actuated once, it will appear to have been actuated several times. This problem is called Key Bouncing.

### **11. What is priority resolver?**

It determines the priorities of the bits set in the Interrupt request register (IRR).The bit corresponding to the highest priority interrupt input is set in the ISR during INTA input.

**12. What is the use of IRR?**

The interrupt request register is used to store all the interrupt levels which are requesting the service.

The eight interrupt inputs sets corresponding bits of the Interrupt Request Register upon the service request.

**13. What is Interrupt service register (ISR)?**

The interrupt service register stores all the levels that are currently being serviced.

**14. What are the operating modes of 8279?**

1. Input modes

Scanned keyboard

Scanned sensor matrix

Strobed input

2. Display modes

Left entry (Type writer mode)

Right entry (Calculator mode)